UCRL-TM-229496

Simulation of Neutron and Gamma Ray Emission from Fission: Fission Library User manual

Fission Library Development Team

Lawrence Livermore National Laboratory

February 23, 2007

1 Introduction

This document describes how to use the general-purpose and extensible software library libFission.a to accurately simulate neutron and gamma-ray emission from fission. The first section describes the fission library interface, the next three sections describe how to run the three popular Monte-Carlo particle transport codes MCNPX [1], Geant4 [2] and COG, using the fission library to sample the fission neutrons and gamma-rays. This user manual complements the physics reference manual "Simulation of Neutron and Gamma Ray Emission from Fission". The software library can be downloaded from
`http://nuclear.llnl.gov/CNP/simulation`.

1.1 Limitations of the fission library

The range of neutron energies for which induced fission neutron multiplicity data were available in the literature spans the range from 0 to 10 MeV, to which corresponds a range of $\bar{\nu}$ values. The sampling of number of neutrons per fission is based on either the incident neutron energy or the nubar corresponding to that energy (depending on the option selected in *setnudist*). When sampling is based on the energy, the neutron multiplicity data at 10 MeV is used for incident neutron energies greater than 10 MeV. This is clearly incorrect, and sampling based on nubar is therefore preferred. When sampling is based on nubar, and the nubar is in the range for which we have multiplicity data from the litterature, that data is used. Outside that range, Terrell approximation is used. Sampling based on nubar is the default for the fission module.

In the case of spontaneous fission, data is only available for the following isotopes: ^{232}Th, ^{232}U, ^{233}U, ^{234}U, ^{235}U, ^{236}U, ^{238}U, ^{237}Np, ^{238}Pu, ^{239}Pu, ^{240}Pu, ^{241}Pu, ^{242}Pu, ^{241}Am, ^{242}Cm, ^{244}Cm, ^{249}Bk, and ^{252}Cf. The 3 Monte-Carlo MCNPX, COG and Geant4 do not emit any particles if a different spontaneous fission isotope is specified.

2 Fission library interface

The interface to the fission library consists of 22 C functions, each of which will be described below:

2.1 void genspfissevt_(int *isotope, double *time)

This function is called to trigger a spontaneous fission. Multiple neutrons and gamma-rays are generated and stored in a stack along with their energies, directions and emission times. The arguments of this function are

isotope: entered in the form ZA (e.g. 94239 for ^{239}Pu)
time: the time of the spontaneous fission

The generated neutrons and gamma-rays, along with their properties will be lost upon the next call to genspfissevt_ or genfissevt_. Therefore, they must be retrieved immediately by the caller using the appropriate functions described below.

2.2 void genfissevt_(int *isotope, double *time, double *nubar, double *eng)

This function is called to trigger a neutron-induced fission. In addition to the arguments above, the fission inducing neutron is characterized by:

nubar: user-specified average number of neutrons emitted per fission (e.g. as tabulated in the cross-section libraries used by the particle transport code)
eng: energy of the neutron inducing fission

Either the average number $\bar{\nu}$ of neutrons emitted per fission or the energy *eng* of the fission inducing neutron will be used to determine the number of neutrons sampled, see function 2.9 below. On the other side, the number of gamma-rays sampled only depends on $\bar{\nu}$. As for genspfissevt_, the generated neutrons and gamma-rays are lost upon subsequent calls to genspfissevt_ and genfissevt_.

2.3 int getnnu_() and int getpnu_()

These functions return the numbers of neutrons and gamma-rays emitted in the fission reaction, or -1 if no number could be sampled in the fission library due to lack of data. The reader is referred to the physics reference manual to find the list of isotopes for which sampling will return positive numbers.

2.4 double getneng_(int *index) and double getpeng_(int *index) double getnvel_(int *index) and double getpvel_(int *index)

These functions return the energies and velocities of the neutrons/gamma-rays.

2.5 double getndircosu_(int *index), double getndircosv_(int *index), double getndircosw_(int *index)

double getpdircosu_(int *index), double getpdircosv_(int *index), double getpdircosw_(int *index)

These 2 families of functions return the direction cosines of the velocity vector on the x, y and z axes for the fission neutrons and gamma-rays.

2.6 double getnage_(int *index) and double getpage_(int *index)

This functions returns the age of the fission neutron/gamma-ray, or -1 if index is out of range. The age returned might be different from the time specified in genfissevt_ and genspfissevt_ for delayed neutrons and gamma-rays, see function setdelay_ 2.7 below. Currently, delayed fission neutrons/gamma-rays are not implemented, so all fission products are emitted promptly.

2.7 void setdelay_(int *delay)

This function is called to enable delayed neutrons and gamma-rays. The argument *delay* is set to

0 (default) for strictly prompt neutrons and photons
1 (n/a) for prompt neutrons, prompt and delayed photons
2 (n/a) for prompt and delayed neutrons, prompt photons
3 (n/a) for prompt and delayed neutrons, prompt and delayed photons

Delayed neutrons and gamma-rays have not yet been implemented in the fission library. This setting has presently no effect on the age sampling. All neutrons and photons are currently emitted promptly (delay=0).

2.8 void setcorrel_(int *correlation)

This function is called to set the type of neutron/gamma-ray correlation. The argument *correlation* is set to

0 (default) for no correlation between neutrons and photons
1 (n/a) for number correlation between neutrons and photons
2 (n/a) for number and energy correlation between neutrons and photons

Correlations between neutrons, photons and their energies, have not yet been implemented. This setting has thus no effect on the outcome of the fission library.

2.9 void setnudist_(int *nudist)

This function is called to set the data to be sampled for the neutron number distributions in neutron-induced fissions. The argument *nudist* can take 3 values:

0 to use the fit to the Zucker and Holden tabulated P(nu) distributions as a function of energy for ^{235}U, ^{238}U and ^{239}Pu.

1 to use fits to the Zucker and Holden tabulated P(nu) distribution as a function of energy for ^{238}U and ^{239}Pu, and a fit to the Zucker and Holden data as well as the Gwin, Spencer and Ingle data (at thermal energies) as a function of energy for ^{235}U.

2 to use the fit to the Zucker and Holden tabulated P(nu) distributions as a function of nubar. The ^{238}U fit is used for the ^{232}U, ^{234}U, ^{236}U and ^{238}U isotopes, the ^{235}U fit for ^{233}U and ^{235}U, the ^{239}Pu fit for ^{239}Pu and ^{241}Pu.

3 (default) to use the discrete Zucker and Holden tabulated P(nu) distributions and correspnding nubars. Sampling based on the incident neutron nubar. The ^{238}U data tables are used for the ^{232}U, ^{234}U, ^{236}U and ^{238}U isotopes, the ^{235}U data for ^{233}U and ^{235}U, the ^{239}Pu data for ^{239}Pu and ^{241}Pu.

For the isotopes not listed above, the Terrell approximation is used. By default, *nudist* is equal to 3.

2.10 void setcf252_(int *ndist, int *neng)

This function is specific to the spontaneous fission of ^{252}Cf. It is called to set the data to be sampled for the (a) Cf252 spontaneous fission number distribution, and (b) Cf252 spontaneous fission neutron energy spectrum. It takes the following arguments:

ndist:

- 0 (default) to sample the number of neutrons from the tabulated data measured by Spencer
- 1 to sample the number of neutrons from Boldeman's data

neng:

- 0 (default) to sample the spontaneous fission neutron energy from Mannhart corrected Maxwellian spectrum
- 1 to sample the spontaneous fission neutron energy from Madland-Nix theoretical spectrum
- 2 to sample the spontaneous fission neutron energy from the Froehner Watt spectrum

By default, both *ndist* and *neng* are set to 0.

2.11 void setrngf_(float (*funcptr) (void)) and void setrngd_(double (*funcptr) (void))

This function sets the random number generator to the user-defined one specified in the argument. If either setrngf_ or setrngd_ are not specified, the default system call srand48 is used. The arguments are random number generator functions that returns variables of type float and double respectively.

3 Geant4

This section describes how to use the fission library with the Monte-Carlo code Geant4.

This physics module has been submitted to the Geant4 collaboration and will appear in a future release. In the meantime, users can compile and link the fission library with Geant4 themselves as described below.

the fission library distribution contains an

The following 3 subsections below describe (a) which c++ classes are necessary to use the fission library, (b) the steps to compile the c++ classes and (c) how to run the executable. The last subsection describes the current limitations.

3.1 Description of c++ classes

The classes PrimaryGeneratorAction, MultipleSource, MultipleSourceMessenger, SingleSource, SponFissIsotope are necessary to generate spontaneous fission neutrons and gammas.

Spontaneous fissions are generated in the PrimaryGeneratorAction class, which derives from G4VUserPrimaryGeneratorA The spontaneous fission source needs to be described in terms of geometry, isotopic composition and fission strength. In the example file, the spontaneous fission source is a volume source emitting in a sphere of radius 3.97 cm centered on the origin. Two isotopes, ^{238}U and ^{235}U fission at the rates of 2.368 and .7475 fissions/second, respectively. Once this information is given, the constructor creates 2 spontaneous fission isotopes, of class SponFissIsotope, and adds them to the source of class MultipleSource. When Geant 4 needs to generate particles, it calls the method PrimaryGeneratorAction::GeneratePrimaries, which firstly sets the time of the next fission based on the fission rates entered in the constructor, and then calls the the method MultipleSource::GeneratePrimaryVertex which determines which one of the two isotopes will fission. This method in turn calls the method SponFissIsotope::GeneratePrimaryVertex for the chosen isotope. It is in this method that the neutrons and photons sampled from the fission library are added to the stack of secondary particles. Sources other than spontaneous fission isotopes can be added to the source of class MultipleSource. For instance, a background term emitting a large number of background gamma-rays can be added, as long as it derives from the class SingleSource. The intensity of that source would be set the same as for the spontaneous fission isotope sources.

For induced fissions, the two classes G4FissLib and G4FissionLibrary are necessary to build the physics process. An instance of the class G4FissLib needs to be added to the PhysicsList to accurately simulate fission processes using the fission library. This is done in PhysicsList via the following code snippet:

```
G4ProcessManager* pmanager = particle->GetProcessManager();
G4String particleName = particle->GetParticleName();

if (particleName == "gamma") {
  (...)
} else if (particleName == "neutron") {
  (...)
  // Fission library model
  G4HadronFissionProcess *theFissionProcess = new G4HadronFissionProcess();
  G4FissLib* theFissionModel = new G4FissLib;
  theFissionProcess->RegisterMe(theFissionModel);
  pmanager->AddDiscreteProcess(theFissionProcess);
  (...)
```

```
} else ...
```

The constructor of G4FissLib does two things. First it reads the necessary fission cross-section data in the file located in the directory specified by the environment variable *NeutronHPCrossSections*. It does this by initializing one object of class G4NeutronHPChannel per isotope present in the geometry. Second, it registers an instance of G4FissionLibrary for each isotope as the model for that reaction/channel. When Geant4 tracks a neutron to a reaction site and the fission library process is selected among all other process for neutron reactions, the method G4FissLib::ApplyYourself is called, and one of the fissionable isotopes present at the reaction site is selected. This method in turn calls G4NeutronHPChannel::ApplyYourself which calls G4FissionLibrary::ApplyYourself, where the induced neutrons and gamma-rays are emitted by sampling the fission library.

3.2 Compilation

Please refer to the `geant` directory in the fission library source code distribution for a complete example and explicit instructions for compiling and linking the fission library with Geant4. Besides setting the usual environmental variables for Geant4, one also needs to set EXTRALIBS to -lFission before building the executable. The fission library libFission.a must be located in a directory specified in the link line. A good place for libFission.a is the directory $G4WORKDIR/tmp/$G4SYSTEM/exec_name The header file Fission.hh must be in the include directory.

3.3 Execution

The environment variable *NeutronHPCrossSections* must point to the G4NDL3.10 directory, where the induced fission cross-sections and data are located.

3.4 Limitations

The induced-fission data available in G4NDL3.10 is scarce. Currently, there are only data files for 7 isotopes of Uranium: ^{232}U, ^{233}U, ^{234}U, ^{235}U, ^{236}U, ^{237}U and ^{238}U. The origin of the data has not been investigated. For other isotopes, induced fission will not emit any particles. The fission library does not have any spontaneous fission data for isotopes other the ones listed in section 1.1.

4 MCNPX

This physics module has been submitted to the MCNPX development team and will appear in a future release. The fission physics included in the library libFission.a is readily available to MCNPX users via a flag on the PHYS:N card. To enable sampling of neutrons and gamma-rays from libFission.a, the 6^{th} entry *fism* of the PHYS:N card should be set to 5:

```
phys:n 100 0 0 -1 -1 5
```

fism=5 is the only MCNPX setting for which gamma-rays are sampled in analog for fission reactions. The fission library can be sampled for both induced and spontaneous fission reactions, the latter only when the *par* flag of the source definition card *sdef* is set to *sf*:

```
sdef par=sf
```

as to specify a spontaneous fission source in MCNPX. In the case of spontaneous fissions, only the isotopes listed above in section 1.1 have data in the fission library. For other spontaneous fission isotopes, no neutrons, nor gamma-rays are emitted.

This *fism* setting has an effect on the standard MCNPX gamma production sampling. By default, MCNPX emits a number of gamma-rays at each neutron collision site, independent on the type of nuclear reaction. This number is sampled from an energy dependent photon yield curve, which is the sum of the production yields from different nuclear reactions. While this method is not accurate on a per reaction basis, this leads to the right number of gammas emitted over a large number of reactions, on an average basis. If these default gamma-rays were sampled with the fission library enabled, the number of gamma-rays would then be off in average. To remediate to this problem, the contribution to the photon yield curve that corresponds to fission reactions is suppressed when libFission.a is in use.

5 COG

This physics module has been submitted to the COG development team and will appear in a future release. The fission library libFission.a can be sampled for induced fissions in COG using the *FISSLIB* keyword in the MIX block of the input deck. This is not the default. The *NUOPTION* can not be used concurrently, it is not compatible with *FISSLIB*.

COG is similar to MCNPX in that it emits a number of gamma-rays at each neutron collision site, and this number is independent on the reaction type. The keyword *NOGAMPRO* can be used to completely turn off this gamma-ray production.

Spontaneous fissions are implemented using a COG user source. A sample user source 'spfiss.F' is available in the COG distribution subdirectory 'usrsor'. Compiling a COG user source is trivial:

```
make -f COGUserlib.make in=spfiss.F
```

Using the right compiler at compile time is important, and if this becomes an issue, a COG developer should be contacted. It is also important to use a COG version that is compatible with user sources and user detectors. Not all COG versions work with user sources/detectors.

An example of input deck using the 'spfiss' spontaneous fission source is located in the 'usrsor' directory. The important lines related to the user source are in the SOURCE block:

```
SOURCE
NPART = 5e4  $ NPART is the sum of spontaneous fission neutrons and photons
$
$ The source below is for a HEU shell (93% enriched in U-235).
$ We neglect here the spontaneous fissions in U-235. The fission rate
$ for 350 g of U-238 is 350[g]*1.36*10E-2[n/g/s] = 4.76 n/s. With
$ spontaneous nubar equal to 2.01, we have
$  4.76[n/s]/2.01[n/fission] = 2.368 fissions/sec
$      name    isotope  strength  xcenter  ycenter  zcenter  Rin  Rout  FissRate
$              (1)      (2)       (3)      (4)      (5)      (6)  (7)   (8)
USRSOR spfiss  92238    1.        0.       0.       0.       1.   3.96  2.368
$
```

NPART is the sum of all source particles, that is both spontaneous fission neutrons and gamma-rays. The line USRSOR has several arguments: The argument under 'name' specify the FORTRAN subroutine to be used a the spontaneous fission source: 'spfiss'. The first numeric argument is the isotope in the form ZA, followed by the source strength (not relevant in this case), the center of the shell (x, y, z), the inner and outer radii and the fission rate in fissions/second. Note the units of the fission rate, fissions/second and not neutrons/second.

References

[1] ”MCNPX Version 2.5.0 User’s Manual”, LA-CP-05-0369, Los Alamos National Laboratory (2005).

[2] *Nuclear Instruments and Methods A* **506**, 250-303 (2003).